特集●ソフトウェア発展

ソフトウェア進化プロセスの統計モデル

玉井 哲雄　中谷 多哉子

オブジェクト指向ソフトウェアのライフサイクルを通じたプロセスをソフトウェア発展の視点から観測すると，多くの興味深い現象が見られる．本論文では，ソフトウェア発展研究のフレームワークを提示し，それに基づいた実証的分析の結果得られた進化・発展パターンについて報告する．

1 はじめに

ソフトウェアは常に変化を続けるが，その変化の速度は近年さらに加速しつつある．このようなソフトウェアの変化の現象の表現の仕方は，さまざまである．いわく,「ソフトウェアの加齢」,「ソフトウェア保守」,「ソフトウェア進化」，あるいは「ソフトウェア発展．ソフトウェアの加齢とは老朽化を暗示する表現である[12]．もっともソフトウェアは，ハードウェアのように磨耗・損傷することはない．老朽化の原因は，保守の反復による構造の劣化や環境変化への不適応である．

そこで，ソフトウェアは保守されるがゆえに老化する，といえよう．保守という用語は，機械工学や電気工学のような既存の工学から輸入されたものである．しかし，上に述べたようにソフトウェアに磨耗・損傷はないので，当初の機能を保つための「保守」は必要がない．ソフトウェアの保守作業には，他の工業製品では保守と呼ばれないデバッグと機能拡張という対照的な目的を持ったものが含まれているという点で，きわめて特殊である．

ソフトウェアが「進化する」あるいは「発展する」(どちらも evolve に対応する語として用いている) という言い方も，実質的にはソフトウェア保守と同じ現象を指しているが，より前向きな響きを持つ．さらに「進化・発展」は自動詞であるが,「保守は他動詞である．すなわち，ソフトウェアは進化・発展するが，人によって保守されるのである．前者の見方はソフトウェアの発展を客観的に観測する手法を示唆するが，後者はソフトウェアが人工物であり，保守は人間によって行われる作業であることを常に意識させる．

ソフトウェア進化の概念がダーウィンの進化論の影響を強く受けるのは，当然である．Belady & Lehman による先駆的な研究[4][8] では，OS/360 の時系列データの分析がなされ，ソフトウェア進化を支配するいくつかの「法則」が見出された．Tamai & Torimitsu による研究[16] では，応用システムの進化プロセスにつき，とくにその作り直し戦略に焦点を当てた分析が行われた．比喩的にいえば，そこで扱われている進化は，1 世代ではなく複数の世代に跨るものであると言える．

この 2 つの研究はいずれも，システムレベルの進化を扱っている．生命進化との対比で言えば，これらは種あるいは個体レベルの進化に対応するといえよう．しかし，現代のソフトウェアは相対的に独立したオブジェクトないしコンポーネントから構成されている．そのようなシステムに関しては，オブジェクトやコン

Statistical Models of Software Evolution Process
Tetsuo TAMAI, 東京大学, The University of Tokyo.
Takako NAKATANI, 有限会社エス・ラグーン, S-Lagoon Co.,Ltd..
コンピュータソフトウェア, Vol., No.(), pp.–.
2002 年 8 月 19 日受付.

ポーネントレベルの進化を考察することが，システムレベルの進化に優るとも劣らず重要であろう．同じクラスの異なるコンポーネント個体が同時に複数のシステム内に生息することや，システムの寿命が尽きてもその中のコンポーネントが別のシステムに移って生き延びることは，通常のことである．おびただしい数のコンポーネント個体群が，全世界に生息し進化を続けている．それらはインターネットを通じて互いに作用し合い，また自由に移動する．ふたたび生命進化との類推を用いれば，システムレベルの進化が種の進化に対応するのに対して，コンポーネントレベルの進化は遺伝子レベルの進化に対応すると言えるだろう．

ダーウィンの進化論の本質は，2 つの仕組みに帰着できる．1 つは複製でありいま 1 つは自然淘汰である．この両者を含むプロセスは，自然現象であろうと人為現象であろうと，進化とみなしてよい．そこで R. Dawkins はミーム (meme) という言葉を，種々の人工概念が普及する社会現象を説明するための遺伝子に対応するものとして提唱した[7]．Dawkins によれば，ミームは文化複製子の単位で，例としては曲，アイディア，惹句，ファッション，壷の作り方，アーチの架け方などがあるという．多くの人がこのミーム概念を発展させた．代表的な例として，S. Blackmore による大胆な展開がある[5]．ソフトウェアコンポーネントの進化は，明らかにミームに基づく進化の特徴を有している．

このような意味でのソフトウェア進化を研究するのには，様々なアプローチがありえよう．有力なものとして，1) 進化プロセスを観察し，進化のパターンや法則を発見する，2) 進化しやすいソフトウェアの開発を支援する計算モデルや言語を設計する，という 2 つの異なる方法がある．われわれは，この 2 つのアプローチの双方を進めてきた．後者についてはすでにいくつかの論文を発表している[14][17][13]．本論文では前者のアプローチについての研究成果をまとめて報告する．

われわれの研究の目的は，オブジェクトやコンポーネントの進化パターンを分析し，ソフトウェア進化モデルを構築することにある．有効なモデルの構築に成功すれば，ソフトウェアの進化プロセスの理解に基盤を提供だきるだけでなく，そのモデルに基づいたソフトウェア環境を開発することにより，長期のソフトウェアプロセスにわたってソフトウェア技術者を支援することが可能となろう．

本論文の構成は以下のようである．第 2 節で，われわれの方法を紹介し，第 1 段階の事例研究の対象としてシステムについて概要を紹介する．第 3 節では，事例研究で得られた進化パターンを示す．第 4 節ではとくに進化データをよく説明する統計モデルに焦点を当て，そのモデルを大規模なソフトウェア進化の事例に適用した結果を報告する．最終節では，この研究で得られた知見と今後の課題について議論する．

2 ソフトウェア進化の分析方法

2.1 測定

分析は常に，測定から始まる．問題は何を測定するかである．クラス数，メソッド数，原始プログラム行数などの簡単な計測量が一般によく用いられる．このような計測量を基に，より複雑なものも含む Chidamber & Kemerer の計測量群[6] のような体系化されたものも広く採用されている[9]．これらの計測量はいずれも，クラスの集合やメソッドの集合上に分布する 1 次元データである．クラス数やメソッド数が大きくなると，データ量は増大し，システム全体に渡る特徴を把握するのは難しくなる．

大量データを扱うための 1 つの有力な方法は，散布図や度数分布図のような視覚的表現を用いることである．このような図により，対象ソフトウェアの特徴を直観的に捉えることが容易にでき，図の形状の時系列的な変化を表示することによってよって，進化プロセスの視覚化が可能になる．

グラフによる表現はこのような直観的な理解には便利であるが，論理的な分析には向かない．そこで，計測データの平均や分散などの基本統計量を求めて全データを代表させ，定量的な分析とすることが通常行われる．しかし，平均と分散という統計量のみでは，観測対象のソフトウェアの製品やプロセスの構造を反映するには単純すぎる．

われわれの発想の要点の一つは，計測されたデータの分布に統計分布モデルを当てはめるところにある．

もし適切な統計分布モデルが見つかれば，計測されたソフトウェアに関するデータは，そのモデルを決定する数個のパラメータの値で特徴づけられることになり，データの構造に関してより深い解釈が可能となろう．

2.2 事例研究

研究の第 1 段階でわれわれは，事例研究による実証的な分析を行った．事例として用いたのは次の 3 つである．

1. **熱交換シミュレーションシステム**
   - システム概要: 多様な熱機器からなるシステムの熱流と温度分布を，シミュレートするシステム．
   - 版数: 4
   - 開発期間: 8ヶ月
   - 開発要員数: 1
   - 開発言語: Visual Smalltalk
   - 規模: 52 クラス（第 4 版）
2. **入金管理システム**
   - システム概要: サービス会社向けの入金管理システムで，顧客からの入金と請求伝票をつき合わせ，消し込み処理を行うもの．
   - 版数: 4
   - 開発期間: 8ヶ月
   - 開発要員数: 1
   - 開発言語: Visual Smalltalk
   - 規模: 62 クラス（第 4 版)
3. **証券管理システム**
   - システム概要: 企業の保有証券データ，すなわち額面価格，買取価格，金利，償還などに関する情報を管理し投資判断を支援するシステム．
   - 版数: 14
   - 開発期間: 3ヶ月
   - 開発要員数: 4
   - 開発言語: Visual Smalltalk
   - 規模: 133 クラス (第 14 版)

各事例で，4 版以上のデータが入手できた．ただし，「版」の意味は最初の 2 つの事例と最後のものでは異なる．第 1 と第 2 のシステムでは，各版は顧客に引き渡され，顧客からは評価と次の版への要求が返されるというプロセスが取られている．一方，第 3 のシステムにおける版は，開発期間中に適宜に置かれたチェックポイントにおけるシステムのスナップショットという性格のものである．これらのシステムの規模はいずれも小さいが，実用に供するために開発されたものであり，実際最初の 2 つのシステムは現在使用されている．

2.3 計測量

われわれは計測すべき量の組を定義し，これら 3 つのシステムの一連の版データに対して，計測を行った．オブジェクト指向システムの計測量についてはすでに多くの提案があり，ここでとくに新たな計測量を考案する必要性は認められなかった．われわれは計測量を，システム，クラス，メソッドの 3 つの層に区分し，システム層についてはクラス数やクラス木の深さ，クラス層についてはメソッド数，インスタンス変数の数，サブクラスの数，メソッド層については原始コード行数，などを計測した．もちろん，1 段下の層のデータの集計値や平均はまた，上の層の計測量となる．たとえば，メソッドの行数の合計や平均は，クラス層の計測量である．

われわれの方法に新しさがあるとすれば，それは測定の基準とした計測量そのものではなく，以下のようなデータの収集と分析の仕方にある．

1. データは 1 つのシステムのある期間内の版の系列全体について収集し，その分析は時系列的に行った．
2. 収集されたデータに関し，基本統計量としての平均や分散だけでなく，その分布の形状にも注目した．

3 観測された進化パターン

このような計測値による定量的分析と相補的なものとして，クラス構造やその他のプログラムの変化を定性的に分析した．また，プロジェクトの文書を調査し，開発者への面談調査も行った．これらを総合し，利用者の要求と開発者の設計意図の変化が，システム

やオブジェクトの進化プロセスにどのように結びつくかを明らかにしようとした．

主な観測結果は次の 4 点にまとめることができる [11].

1. 基本統計量と分布形は，時間の推移に対し相対的に安定している．
2. 一方で，例外的に大きな特異値を持つデータが存在することがある．これらは設計上の不具合か例外的な設計判断を反映していると見られる．
3. 多くのデータは時間軸上で右上がりに増大する傾向を示すが，その変化は連続的ではない．急激な上昇の期間と変化の遅い期間とがある．非連続的な変化は，アーキテクチャレベルの変更が行われたことを示す場合が往々にしてある．
4. クラス木を特徴づける特有の計量の存在が発見された．

以下ではこのうち，とくに 1 番目と 4 番目の知見について，詳しく論じる．

3.1 統計モデルの安定性

オブジェクトシステムの規模に関しては，伝承的なデータが知られている．Smalltalk プログラマとして 100 万行以上のコードを書いてきたという A. Aoki は [2]，自ら開発したあらゆるシステムやライブラリの実績から，クラス当たりの平均メソッド数は 20，メソッド当たりの平均行数は 10，したがってクラス当たりの平均行数は 200 だといっている．さらにこの値は，自分の開発したプログラムのみならず，提供されている標準ライブラリや他の組織で開発されたコードでも変わらないという．

面白いことに，われわれの事例でもこの値が確認された．表 1 に示すデータは，この観測を裏付けるものである．これらの値は時間 (版) によらず，またシステムによらずほぼ一定である．

図 1 と図 2 は，典型的な規模データの度数分布図を示したものである．これを見ると，異なる版だけでなく異なるシステム間でも平均値がほぼ同じであるばかりか，分布形も共通していることがわかる．これらのグラフから，規模データの分布を説明するような共通な統計モデルの存在が示唆される．

表 1　熱交換シミュレーションシステムの基本データ

クラス当たりのメソッド数

版	1	2	3	4
平均	15.1	19.4	19.7	18.3
標準偏差	10.3	16.5	19.4	19.9

メソッド当たりの行数

版	1	2	3	4
平均	8.1	8.5	9.1	9.4
標準偏差	10.8	16.0	19.5	21.5

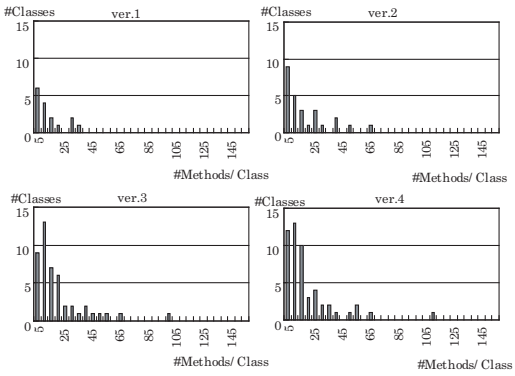

図 1　熱交換シミュレーションシステムのクラス当たりメソッド数の度数分布図

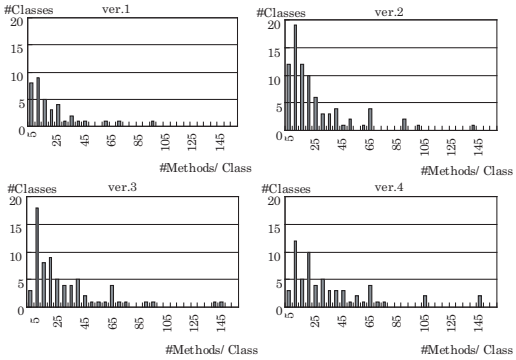

図 2　入金管理システムのクラス当たりメソッド数の度数分布図

一見したところでは，ポアソン分布が適合するように見える．しかし，ポアソン分布の当てはめを試行してみると，適合しないことが簡単に判明する．幾何分

布を試してみても，うまくいかない．そこで注目したのが，負の 2 項分布である．負の 2 項分布が有望な理由として，次の 2 点を挙げることができる．

1. ポアソン分布と比べ，平均値が同じ場合，分散が大きくなる．図 1 や図 2 のような右に長い裾野を持つ度数分布に対しては，分散の大きい負の 2 項分布の方が適合する可能性がある．
2. 負の 2 項分布は，ベルヌーイ試行において，ある事象があらかじめ定められた回数だけ起こったときの，試行系列の長さの分布として定義される．すなわち，負の 2 項分布は元来,「長さ」の分布という性格を持つので，コードの長さの分布を説明するのに適している可能性がある．

図 3 は，入金管理システムのコード行数データに対し，曲線の当てはめを行った結果を示す．見た目にはよく適合しているようだが，Pearson の適合度検定を適用してみると，適合性は必ずしも有意とはいえないという結果となった．

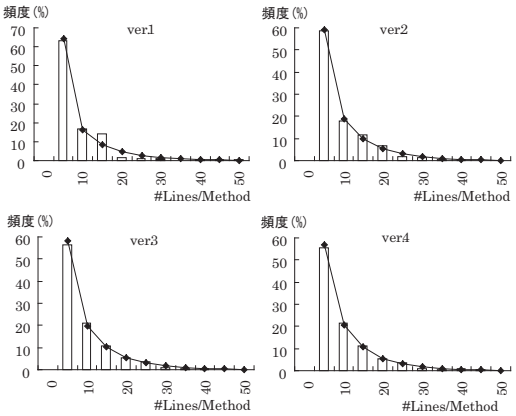

図 3　メソッド当たり行数の分布への負の 2 項分布モデルの当てはめ

このモデルの適用を精緻化するために母集団を分割し，同じクラス木に属するクラスを集めた単位に分けてモデル適用を試みた．次節に述べるように，同じクラス木に属するクラスは，それぞれの木に固有な均質の性質を持つことが分かっている．したがって，各木に属するクラス集合にモデルを当てはめた場合，より適合することを期待できる．実際，負の 2 項分布が適合するという仮説は，ここで取り上げた 3 つのシステムのほとんどのクラス木に対し，5%レベルの χ^2 検定で棄却されなかった．

負の 2 項分布モデルについては，後の節で改めて詳しく扱う．

3.2　クラス木の特徴づけ

予想されるように，クラス当たりのコード行数とメソッド数は強い相関を持つ．実際，3 つのシステムについてクラスのコード行数とメソッド数の相関を統計的に検定した結果，有意と認められた．

図 4 の上段左に，熱交換シミュレーションシステムの関して，この 2 つの値を両軸にとった散布図が描かれている．ここでサンプルは，このシステムの 4 つの版すべてのものを集めて表示している．すぐに気づくように，この図では相関を示す直線が複数本識別される．調べてみると，これらの直線はそれぞれ同じクラス木に属するクラス集合に対応したものであることが分かる．図 4 の他の 3 つの散布図は，これを 3 つのクラス木に分けて表したものである．

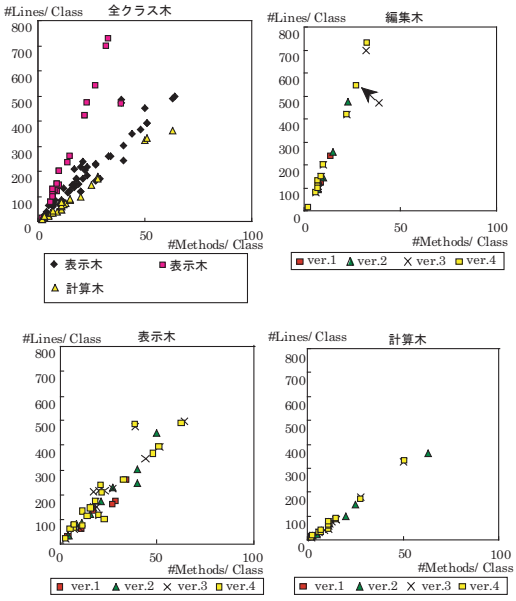

図 4　熱交換シミュレーションシステムの行数対メソッド数の散布図

きわめて注目すべき現象が，編集木（上段右) の散布図に見られる．図中の矢印は，この木に属するあるクラスの第 3 版における値から第 4 版における値への移動を示している．このクラスの第 3 版における行数とメソッド数に対応した座標点は，直線から大きく外れた例外値であるが，第 4 版では「正常な」値に戻っている．この現象は，同じクラス木の行数/メソッド数の値，あるいは両測定値間の回帰係数の値が，かなり強力な拘束力を持つものであることを示唆するものである．

この木に固有な値の持つ意味をさらに探求するために，以下の 3 つの仮説を立て，統計的検定を行った．

1. **異なるクラス木の回帰係数の値は異なる．**
   この仮説は，異なる木の回帰係数が等しいという帰無仮説が棄却されることにより，検証された．3 システムすべてでこの仮説が支持された．
2. **クラス木の回帰係数は進化の過程で不変である．**
   この仮説は，異なる版における同じクラス木の回帰係数が等しいという帰無仮説が，棄却できないことにより検証された．
3. **クラス木の回帰係数は異なるプログラマ間でも安定している．**
   われわれのデータでは，1 つのクラス木に属するクラスを，複数のプログラマで分けて開発している例は，1 例しかなかった．そのケースでは，両者の間での回帰係数に統計的に有意な違いは認められなかった．ただ，これは 1 例でしかないので，確定的な結論とはいえない．

これらの知見から，各クラス木には，開発者が暗に仮定しているある種の設計基準が存在することが，示唆される．このような計測量を監視して開発者に示すことは，よい開発支援となるであろう．

4 分布モデル

4.1 負の 2 項分布

オブジェクト指向以前の昔から，モジュール別のコード行数のようなプログラム規模を表すデータは，もっとも基本的な統計量として収集されてきた．それらの度数分布図も度々描かれてきたが，不思議なことに，これまでそのようなデータの分布がなんらかの統計モデルに従うのではないかという議論は，われわれの知る限りではなされてこなかったようだ．

図 2 が示すように，オブジェクトのクラスやメソッドの規模データは，以下のような共通の特徴を持つ．

1. 度数分布の山は 1 つで，全区間の左側にある．
2. 平均値は山よりは右，区間の中央よりは左にある．
3. 分布形右側は，長い裾野を持つ．

第 3.1 節で述べたように，多くのケースで負の 2 項分布が計測値とよく一致していることが観測されている．負の 2 項分布は次のように定義される．事象 S が起こるかどうかを観察する繰り返し試行において，各試行が互いに独立で，S の起こる確率 p が時間に依らず一定であるとき，それはベルヌーイ試行と呼ばれる．S がちょうど k 回起こったときの試行系列の長さを x とすると，x の確率関数は，

$$P(x) = \left(\begin{array}{c} x-1 \\ k-1 \end{array} \right) p^k (1-p)^{x-k}, \qquad (1)$$

で与えられる．

この x の確率分布を負の 2 項分布と呼ぶ．この確率モデルは 2 つのパラメータ，p と k で定められる．

負の 2 項分布に従う x の期待値は，

$$E(x) = k/p, \qquad (2)$$

で与えられ，分散は

$$V(x) = k(1-p)/p^2, \qquad (3)$$

で与えられる．

負の 2 項分布に従うことが予想されるサンプルデータが与えられたとき，パラメータの推定値 $\hat{p}$ と $\hat{k}$ はそれぞれ，

$$\hat{p} = \bar{x}/(s^2 + \bar{x}), \qquad (4)$$

と

$$\hat{k} = \bar{x}^2/(s^2 + \bar{x}), \qquad (5)$$

で与えられる．ここで，$\bar{x}$ はサンプルデータの平均値，s^2 は不偏分散である．

負の 2 項分布は実データとの適合性がよいだけでなく，ソフトウェア開発プロセスのモデルとしての解釈が可能な点に意義がある．コードの長さが決定されるプロセスは，次のような確率プロセスとして解釈することができる．プログラマによるプログラミング作業を，第 3 者が観察しているとする．観察者の

目には，プログラミングは，文（または行) あるいはメソッドの無作為の選択の繰り返しに見える．ある定まった数 (パラメータ k に対応) の特定の性質を持った文が選ばれると，メソッド（あるいはクラス) が完結する．ランダムに選ばれた文（あるいはメソッド）が特定の性質を持つ確率 (p に対応) は一定である．

この解釈に従うと，k が大きければ，問題領域や開発環境・組織で規定される慣例，スタイル，制約などによって要求される特定の文（あるいはメソッド) の集合から，より多くを選ばなければならない．また，p が大きければ，そのような特定の文（あるいはメソッド) を，そのような制約から自由な文との対比で，相対的により多く選ばなければならないことを意味する．したがって，一般的に，ソフトウェア設計あるいはプログラミングがより自由な場合，すなわちプログラマの裁量の余地が大きい場合，k と p は小さくなる傾向にあり，設計やプログラミングががよりパターン化，規律化されている場合は，それらが大きくなる傾向にあるといえよう．

4.2 モデルのパラメータの進化

2 つのパラメータ p と k でモデルが完全に決定されるので，この両者の値は平均値と分散よりも豊かな情報を含んでいる．大量のデータの有する構造が，(p, k) の作る 2 次元空間の 1 点で表されることになり，複数の版に渡るデータの動きを追跡するのに都合がよい．

図 5 は，熱交換シミュレーションシステムのクラス木ごとのパラメータ推定値をプロットしたものである．図の中で矢印は，版の進行に沿った推移方向を表す．同様なグラフを入金管理システムに対して描いたのが図 6，証券管理システムに対して描いたのが図 7 である．

これらのグラフから，以下のようなことが分かる．

1. 両パラメータの間には，強い線形関係がある．各点がのることが推定される直線は (p, k) 平面の原点を通過するので，関係
$$k = mp, \qquad (6)$$
の成立が予想される．ここで，m は定数係数である．

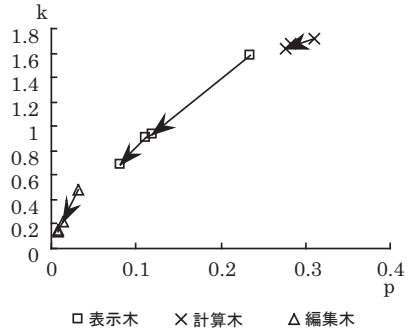

図 5 熱交換シミュレーションシステムのパラメータ (p,k) の動き

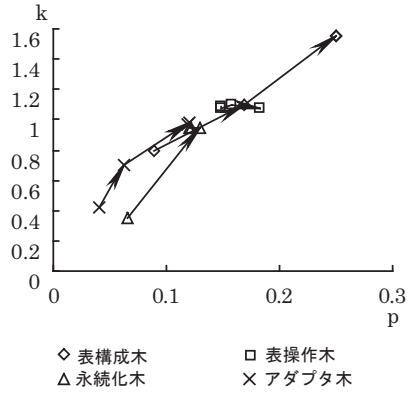

図 6 入金管理システムのパラメータ (p,k) の動き

等式 (2) の $E(x) = k/p$ という関係を思い起こすと，m は x の期待値にあたることが分かる．すなわち，メソッド数の平均は，版に依らず一定であることを意味する．

2. この線形関係により，k の値が大きくなれば，p の値も大きくなり，逆に一方が小さくなれば他方も小さくなる．k と p が大きいことは，パターン化されたコーディング，強い規約，構造の一様化されたプログラムを意味し，k と p が小さいことは，プログラミングのより大きな自由度を表す．
3. 図の矢印の進行方向を見ると，熱交換シミュレーションシステムでは k や p は時間とともに減少し，入金管理システムでは時間とともに増大している．また，証券管理システムではどちらともいえない．熱交換シミュレーションシステムでは，

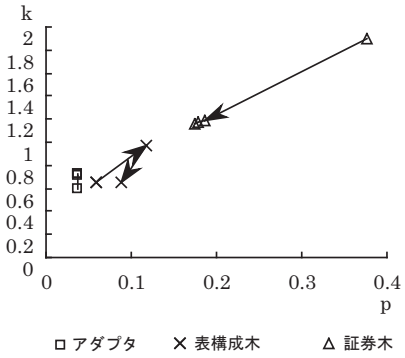

図 7　証券管理システムのパラメータ (p,k) の動き

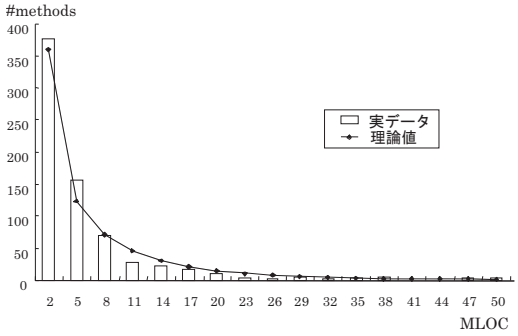

図 8　「幾何」木に対する負の 2 項分布の当てはめ

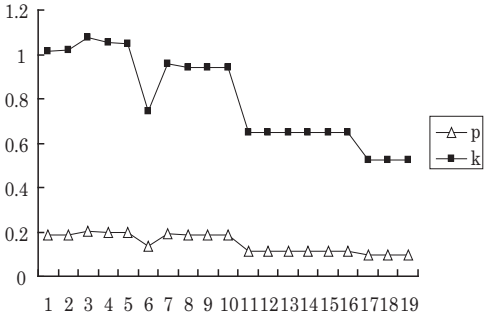

図 9　p と k の時間変化

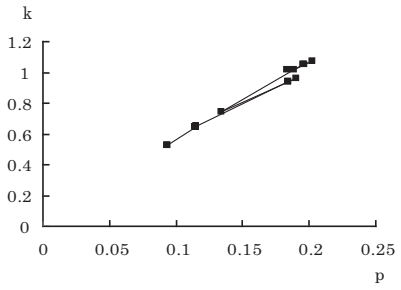

図 10　p と k の (p,k) 平面上の軌跡

版を改めるごとに利用者の要求に応じて新たなモジュールを追加してきており，一方，入金管理システムでは版が進むにつれてソフトウェア開発者がシステムを再構築してきた．この事実は，上に述べた k と p の意味づけとよく対応する．

4.3　より大きな事例研究

より大きな事例研究として，5 年の開発期間のうちに 360 の版を出したグラフィックライブラリ，Jun, を取り上げた[1][3]．このソフトウェアもやはり Smalltalk で書かれ，すべての版のデータが保存されている．

われわれは，その第 93 版から第 206 版までの中で利用者に配布された 19 の主要な版を取り出し，分析した．その開発期間は 14ヶ月強である．この間に，システム規模はほぼ倍増した．クラス数は 195 から 390 に，メソッド数は 4532 から 7708 に，プログラム行数は 25542 から 47736 に増大している．

システムを構成する 5 つの主なクラス木の 19 版に渡るデータに対して，負の 2 項分布モデルの当てはめを試みた．図 8 に,「幾何」と名づけられたクラス木のクラス当たりメソッド数に対し負の 2 項分布を当てはめた例を示す．このモデルの適合性は統計的に検証されている．

図 9 は，19 版の「幾何」木に対して推定されたパラメータ値 p と k の動きを示している．図 10 は同じ推移を，(p, k) 平面で示したものである．

前の事例研究で観測されたように，この例でも p と k は明らかな線形関係を有する．このことは，図 11 に示すように，p と k の代わりに mp と k をプロットすることで，より鮮明になる．ここで m は推定平均値である．

次に観測されることとして，p と k の値が比較的に安定した 4 つの区間の存在を挙げることができる．それらの区間は，版の 1–5, 7–10, 11–16, および 17–19 である．また，3 つの大きな下向きの飛びが 5–6, 10–11,

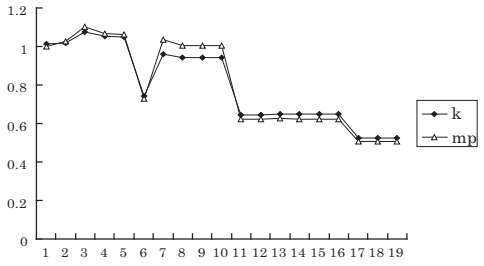

図 11 mp と k の時間変化

16–17 に，2 つの大きな上向きの飛びが 2–3, 6–7 に観測される．

この間，システムの規模は，クラス数，メソッド数，コード行数のいずれで見ても，単調に増加している．それに対し，(p, k) のレベルは，総じて減少傾向にあるものの，その動きは 2–3 や 6–7 に上昇が見られるように単調ではない．この (p, k) の上昇期間は，再構成 (refactoring) が行われた時期で，その下降期間は機能拡張の時期であると見なすことができる [10]．したがって，(p, k) は機能拡張にしたがって全体的に減少傾向にあるが，再構成のための上昇時期が時折交代して現れるという過程の存在を認めることができる．

5 議論

分布モデルとその解釈を用いることにより，システムの進化プロセスを論理的にも視覚的にも洞察できることを示した．規模データの平均値の安定性は，単に計測データの平均値を取るだけで直接的に発見できたはずだという議論もありえよう．また，(p, k) の上昇と下降はほぼ分散の減少と増加に対応する（しかし常にではない）ので，機能拡張と再構成の期間の議論も，分散を追うだけでできるのではないかという主張も考えられる．しかし，パラメータの p と k は平均と分散を定めるだけでなく，分布の形そのものを規定する．したがって，システムの変化パターンを時系列的に追跡する際に，これらはより豊富な情報を与えるものと言える．たとえば，観測した実データの中に，この分布形から大きく外れるものがあれば，それを捉えて分析することもできよう．また，k の値を具体的なプログラミングの規約や制約と結びつけて論じることも，可能であろう．

これらの結果を，いかに活用すべきだろうか．これまでは過去の版データの分析のみを行ってきた．しかし，ソフトウェアの進化プロセスを通じて実時間で監視したとすれば，開発者に観測結果をフィードバックすることができよう．もし開発者の意図と観測された現象の間に乖離が見つかった場合，その開発作業を再考するきっかけを与えることになりうるだろう．同じように，データを監視し，視覚化し分析することにより，種々の変則な状況が発見できる可能性がある [15]．

これまで分析したいずれのシステムでも，パラメータ p と k は明確な線形関係を示したが，その値はシステムごとに，また同じシステムでもクラス木ごとに，異なるものであった．設計やプログラミングの制約の強弱は p と k の水準の違いの一部を説明するが，それ以外の要因が影響していることも予想される．その要因が何であるかを探求することは，われわれの今後の研究課題の 1 つである．

参考文献

[1] A. AOKI ET AL. Jun home page. http://www.sra.co.jp /people /aoki /Jun/.

[2] AOKI, A. Smalltalk textbook. http: //www.sra.co.jp /people /aoki /SmalltalkTextbook /index.html.

[3] AOKI, A., HAYASHI, K., KISHIDA, K., NAKAKOJI, K., NISHINAKA, Y., REEVES, B., TAKASHIMA, A., AND YAMAMOTO, Y. A case study of the evolution of Jun: an object-oriented open-source 3D multimedia library. In *International Conference on Software Engineering (ICSE'01)* (2001), pp. 524–533.

[4] BELADY, L. A., AND LEHMAN, M. M. A model of large program development. *IBM Systems Journal 15,* 3 (1976), 225–252.

[5] BLACKMORE, S. J. *The Meme Machine*. Oxford University Press, 1999.

[6] CHIDAMBER, S. R., AND KEMERER, C. F. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering 20,* 6 (1994), 476–493.

[7] DAWKINS, R. *The Selfish Gene*. Oxford University Press, 1976.

[8] LEHMAN, M. M., AND BELADY, L. A. *Program Evolution: Processes of Software Change*. Academic Press, 1985.

[9] LORENZ, M., AND KIDD, J. *Object-Oriented Software Metrics*. Prentice-Hall, 1994.

[10] NAKATANI, T. Quantitative observations on object evolution. In *International Workshop on Principles of Software*

Evolution (IWPSE'01) (Vienna, Austria, Sept. 2001).

[11] NAKATANI, T., TAMAI, T., TOMOEDA, A., AND MATSUDA, H. Towards constructing a class evolution model. In *Asia-Pacific Software Engineering Conference* (Hong Kong, December 1997), pp. 131–138.

[12] PARNAS, D. Software aging. In *16th International Conference on Software Engineering* (Sorrento, Italy, May 1994), pp. 279–287.

[13] TAMAI, T. Objects and roles: modeling based on the dualistic view. *Information and Software Technology 41*, 14 (1999), 1005–1010.

[14] TAMAI, T. Evolvable programming based on collaboration-field and role model. In *International Workshop on Principles of Software Evolution (IWPSE'02)* (Orlando, Florida, May 2002), ACM, pp. 1–5.

[15] TAMAI, T., AND NAKATANI, T. An empirical study of object evolution processes. In *International Workshop on Principles of Software Evolution (IWPSE'98)* (Kyoto, Oct. 1998), pp. 33–37.

[16] TAMAI, T., AND TORIMITSU, Y. Software lifetime and its evolution process over generations. In *Proc. Conference on Software Maintenance – 1992* (Orlando, Florida, November 1992), pp. 63–69.

[17] UBAYASHI, N., AND TAMAI, T. An evolutional cooperative computation based on adaptation to environment. In *Proc. Asia Pacific Software Engineering Conference '99* (Takamatsu, Japan, Dec. 1999), IEEE Computer Society, pp. 334–341.